

Programming Principles in Python (CSCI 503/490)

Introduction

Dr. David Koop

Python Experience?

Programming Principles?

Why Python?

Productivity

Libraries, Libraries, Libraries

What about speed?

Why Principles?

Administrivia

- Course Web Site
- TA: Dalia Khaizaran
- Syllabus
 - Academic Integrity
 - Accommodations
- Assignments
- Tests: 2 (Feb. 18, Apr. 1) and Final (May 6)
- Course is offered to both undergraduates (CS 490) and graduates (CS 503)
 - Grad students have extra topics, exam questions, assignment tasks

Schedule

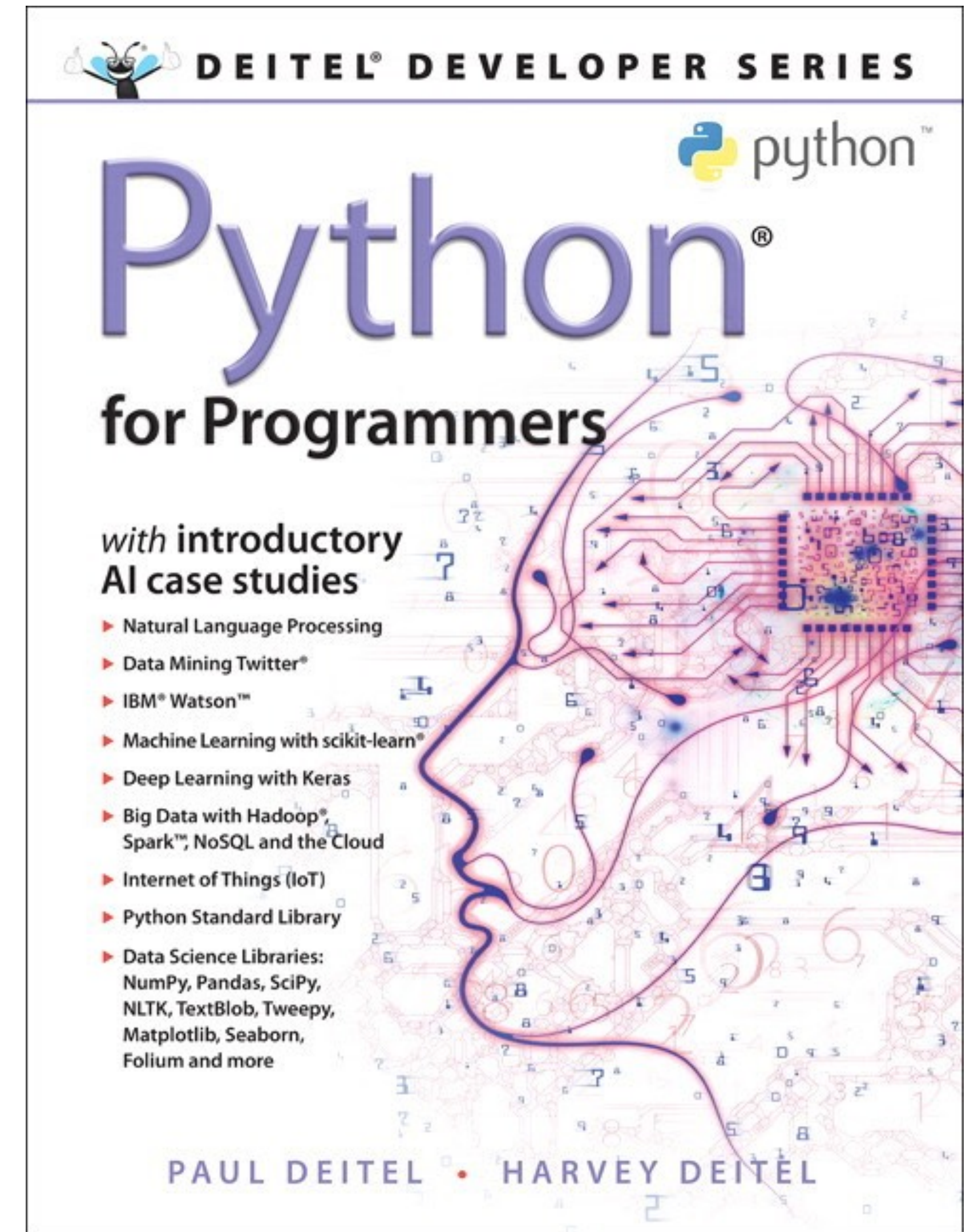
- Lectures are 9:30-10:45am MW in PM 203
 - Better for learning if you are engaged
 - **Ask questions**
 - Please advise me of any issues
 - Short quizzes may be given at the beginning of class
- Any changes will be announced as soon as possible
- Slides will be posted to the course website

Office Hours & Email

- TA office hours will be held in person in TA Offices (PM 362)
 - TuTh: 9:00am-12:00pm
- Prof. Koop's office hours will be held in person in PM 461
 - M: 1:45-3:00pm, W: 10:45am-12:00pm, or by appointment
 - You do not need an appointment to stop by during scheduled office hours,
 - If you wish to meet virtually, please schedule an appointment
 - If you need an appointment, please email me with **details** about what you wish to discuss and times that would work for you
- Many questions can be answered via email. **Please consider writing an email before scheduling a meeting.**

Course Material

- Textbook:
 - Recommended: Python for Programmers
 - Good overview + data science examples
- Many other resources are available:
 - <https://wiki.python.org/moin/BeginnersGuide>
 - <https://wiki.python.org/moin/IntroductoryBooks>
 - <http://www.pythontutor.com>
 - <https://www.python-course.eu>
 - <https://software-carpentry.org/lessons/>



Course Material



- Software:
 - Jupyter Notebook: Web-based interface for interactively writing & executing Python code
 - JupyterLab: An updated web-based interface that includes the notebook and other cool features
 - JupyterHub: Access everything through a server
 - Install using conda/mamba, pixi, uv

Online Students

- Same course material
- In-person lectures are recorded and shared online (same day)
- All deadlines are the same as for in-person students
- Quizzes conducted online
- Instructor and TA will be available for office hours online
- Exams will be conducted via Blackboard: Instructor may conduct them synchronously (Zoom/Teams) and/or using Respondus LockDown browser

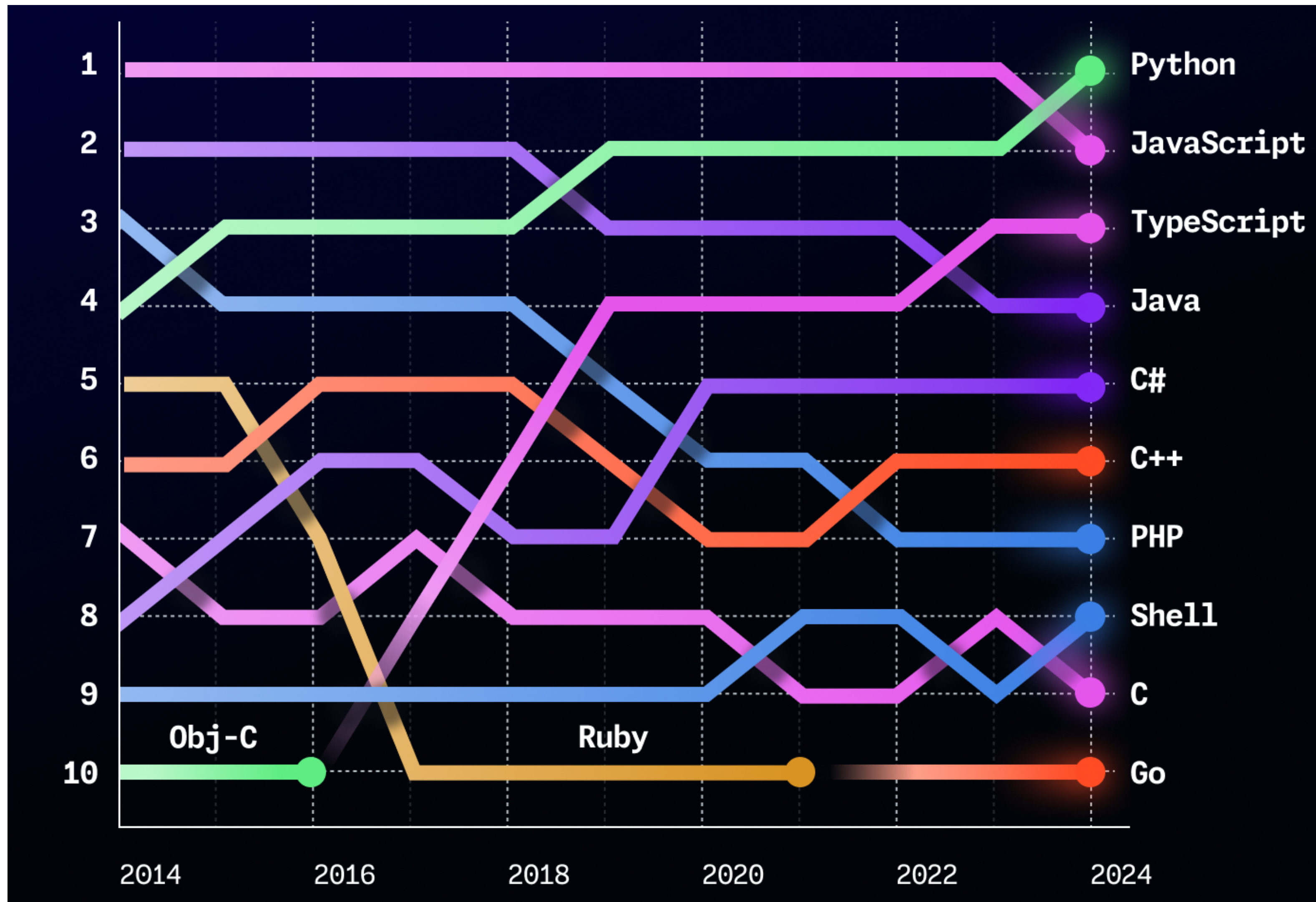
Academic Integrity

- **Do not cheat!**
- You will receive a **zero** for any assignment/exam/etc. where cheating has occurred
- You will **fail** the course if you cheat more than once
- Misconduct is **reported** through the university's system
- You **may** discuss problems and approaches with other students
- You **may not** copy or transcribe code from another source
- **Do not** use artificial intelligence solutions to write code

Python

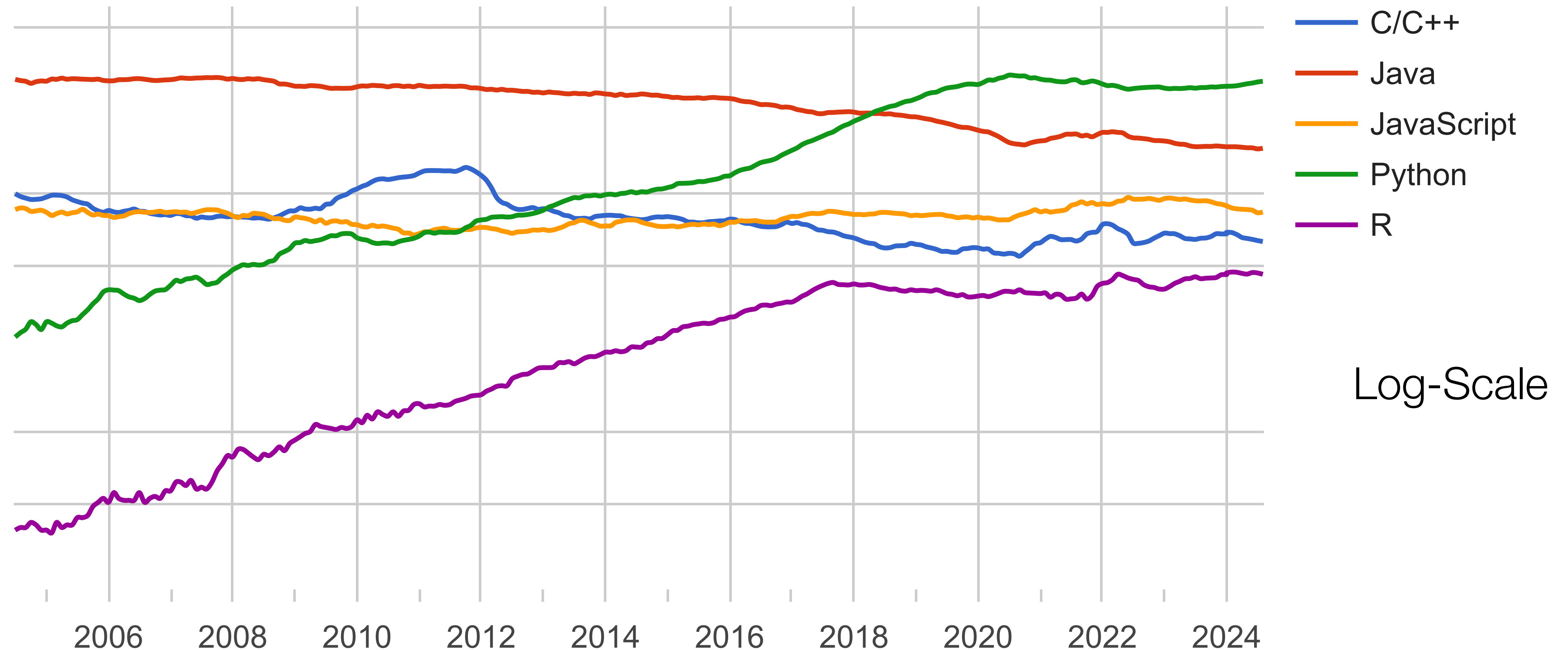
- Started in December 1989 by Guido van Rossum
- “Python has surpassed Java as the top language used to introduce U.S. students to programming...” ([ComputerWorld](#), 2014)
- Python is also a top language for data science
- High-level, interpreted language
- Supports multiple paradigms (OOP, procedural, functional)
- Help programmers write **readable** code, use less code to do more
- Lots of libraries for python
- Designed to be extensible, easy to wrap code from other languages like C/C++
- Open-source with a large, passionate community

Python the #1 Programming Language in 2024



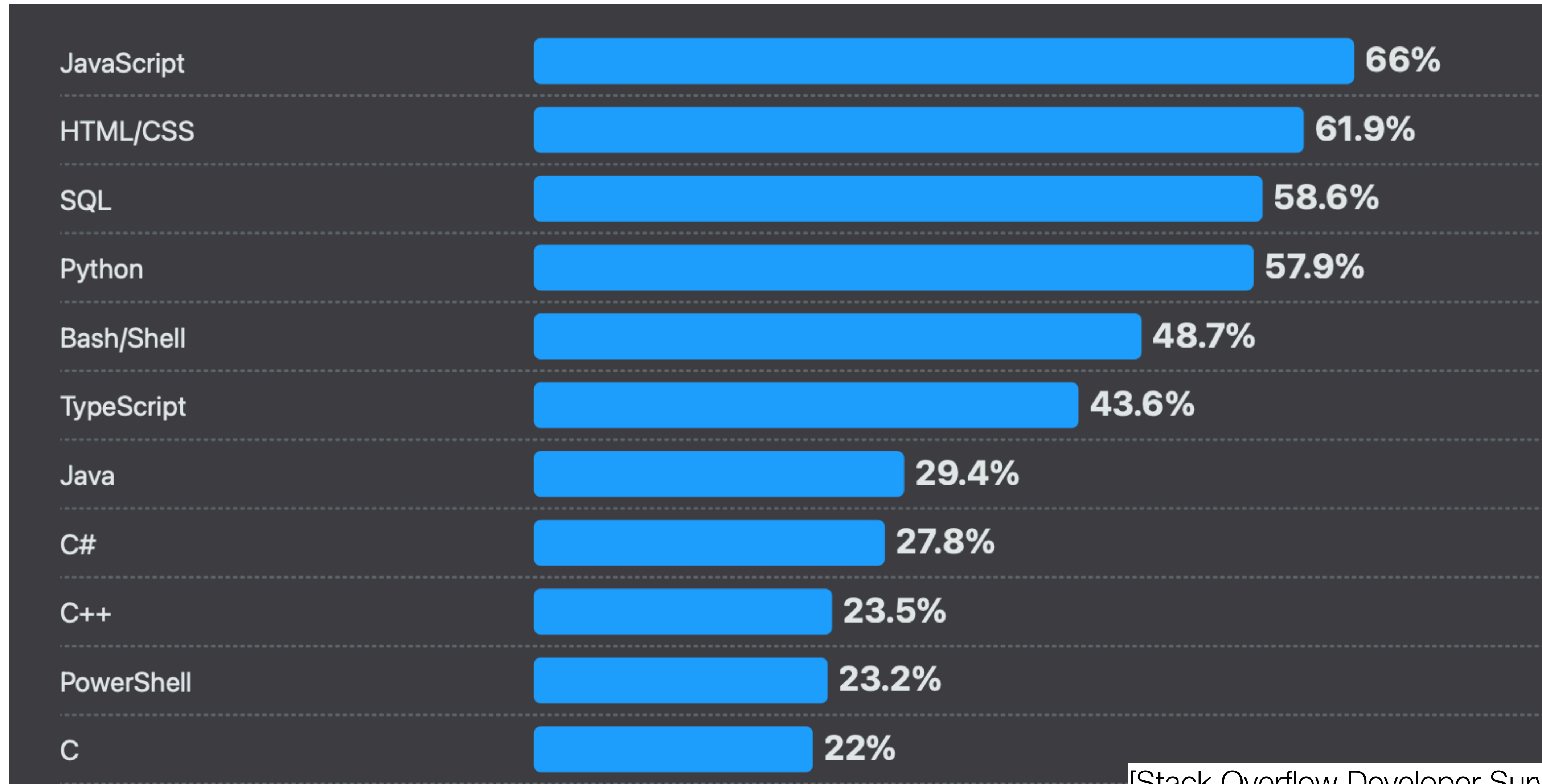
[GitHub]

Even Wider Gap in Google Tutorial Searches



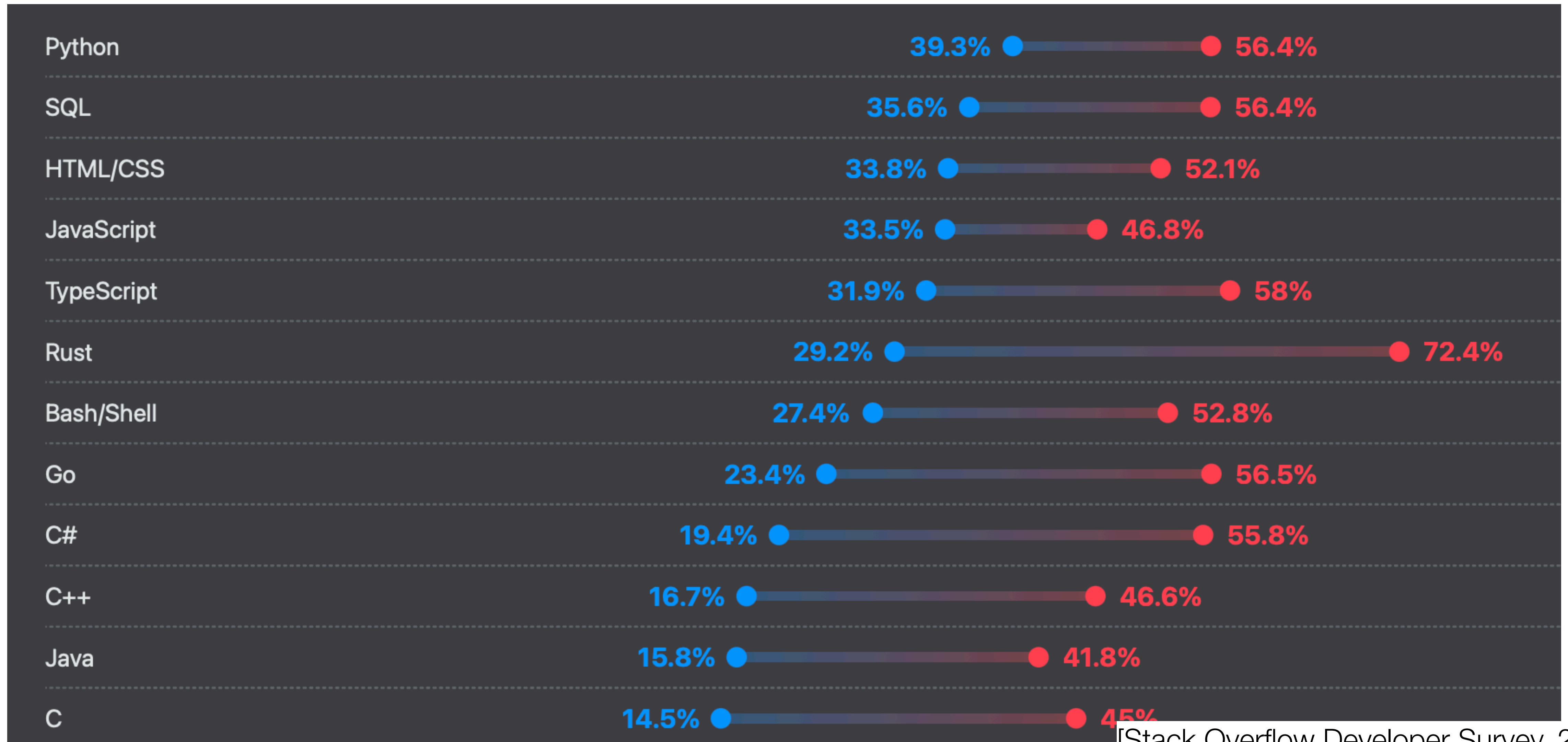
[Popularity of Programming Language]

StackOverflow Language Usage



[Stack Overflow Developer Survey, 2025]

StackOverflow Admired and Desired



[Stack Overflow Developer Survey, 2025]

Modes of Computation

- Python is **interpreted**: you can run one line at a time without compiling
- Interpreter in the Shell
 - Execute line by line
 - Hard to structure loops
 - Usually execute whole files (called scripts) and edit those files
- Notebook
 - Richer results (e.g. images, tables)
 - Can more easily edit past code
 - Re-execute any cell, whenever

Python Differences

- Dynamic Typing
 - A variable does not have a fixed type
 - Example: `a = 1; a = "abc"`
- Indentation
 - Braces define blocks in Java, good style is to indent but not required
 - Indentation is critical in Python

```
z = 20
if x > 0:
    if y > 0:
        z = 100
else:
    z = 10
```

JupyterLab and Jupyter Notebooks

The screenshot displays the JupyterLab environment. On the left, a sidebar contains a 'Files' panel with a list of notebooks and files, including 'Data.ipynb', 'Fasta.ipynb', 'Julia.ipynb', 'Lorenz.ipynb' (selected), 'R.ipynb', 'iris.csv', 'lightning.json', and 'lorenz.py'. Below this are 'Running' and 'Commands' panels. The main area is divided into three panes: a top pane showing the 'Lorenz.ipynb' notebook with text and equations, a bottom-left pane showing the 'Output View' with sliders for parameters and a 3D plot, and a bottom-right pane showing the 'lorenz.py' source code.

Files Panel:

Name	Last Modified
Data.ipynb	an hour ago
Fasta.ipynb	a day ago
Julia.ipynb	a day ago
Lorenz.ipynb	seconds ago
R.ipynb	a day ago
iris.csv	a day ago
lightning.json	9 days ago
lorenz.py	3 minutes ago

Notebook Content:

In this Notebook we explore the Lorenz system of differential equations:

$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$

Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two points, called attractors.

In [4]: `from lorenz import solve_lorenz`
`t, x_t = solve_lorenz(N=10)`

Output View:

sigma: 10.00
beta: 2.67
rho: 28.00

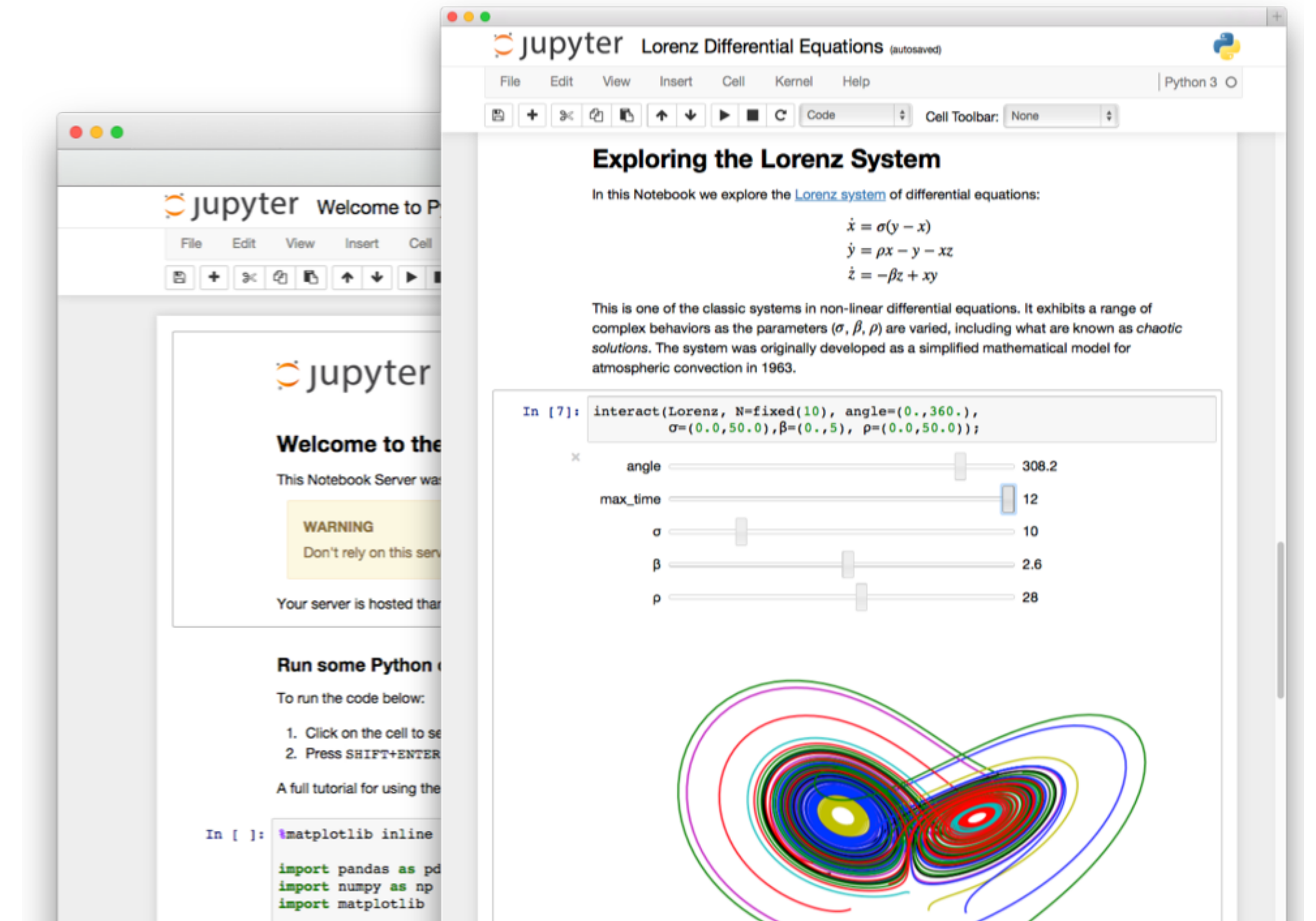
lorenz.py Source Code:

```
9 def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
10     """Plot a solution to the Lorenz differential equations."""
11     fig = plt.figure()
12     ax = fig.add_axes([0, 0, 1, 1], projection='3d')
13     ax.axis('off')
14
15     # prepare the axes limits
16     ax.set_xlim((-25, 25))
17     ax.set_ylim((-35, 35))
18     ax.set_zlim((5, 55))
19
20     def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
21         """Compute the time-derivative of a Lorenz system."""
22         x, y, z = x_y_z
23         return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]
24
25     # Choose random starting points, uniformly distributed from -15 to 15
26     np.random.seed(1)
27     x0 = -15 + 30 * np.random.random((N, 3))
28
```

[JupyterLab Documentation]

Jupyter Notebooks

- Display rich representations and text
- Uses Web technology
- Cell-based
- Built-in editor
- GitHub displays notebooks



[Jupyter]

Jupyter Notebooks



- An interactive programming environment
- Runs in your web browser
- Displays results (even interactive maps) inline
- Originally designed for Python
- Supports other languages, too
- You decide how to divide code into executable cells
- Shift+Enter (or the "play" button) to execute a cell

JupyterLab Notebooks

- Starts with a directory view
- Create new notebooks using the Launcher (+ icon on the left)
 - New notebooks have the name "Untitled"
 - File → Rename Notebook... (or right-click) to change the name
- Save a notebook using the command under the File menu
- Shutting down the notebook requires quitting the kernel
 - Web browser is **interface** to display code and results
 - **Kernel** runs the code: may see messages in a console/terminal window
 - Closing the browser window does not stop Jupyter
 - Use File → Hub Control Panel → Stop My Server to reset on tiger

JupyterLab Notebooks

- Open a notebook using the left panel like you would in a desktop view
- Past results are displayed—does not mean they are loaded in memory
- Use "Run All" or "Run All Above" to re-execute past work
 - If you shut down the kernel, all of the data and variables you defined need to be redefined (so you need to re-run all)
 - **Watch Out—Order Matters:** If you went back and re-executed cells in a different order than they are shown, doing "Run All" may not produce the same results!
- Edit mode (green) versus Command mode (blue == **Be Careful**)

JupyterLab Notebooks

- Can write code or plain text (can be styled Markdown)
 - Choose the type of cell using the dropdown menu
- Cells break up your code, but all data is **global**
 - Defining a variable `a` in one cell means it is available in **any** other cell
 - This includes cells **above** the cell `a` was defined in!
- Remember **Shift+Enter** to execute
- Enter just adds a new line
- Use `?<function_name>` for help
- Use Tab for **auto-complete** or suggestions
- Tab also indents, and Shift+Tab unindents